

Содержание

Лабораторная работа № 1	2
Лабораторная работа № 2	6
Лабораторная работа №3	9
Лабораторная работа № 4	12
Лабораторная работа №5	15
Лабораторная работа № 6	19
Лабораторная работа № 7	22
Лабораторная работа №8	25

Лабораторная работа № 1

Среда программирования . Интерфейс программы.
Компоненты Button, Label

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Внешний вид среды программирования отличается от многих других из тех, что можно увидеть в Windows.

Главные составные части среды программирования

Ниже перечислены основные составные части :

1. Дизайнер Форм (Form Designer)
2. Окно Редактора Исходного Текста (Editor Window)
3. Палитра Компонент (Component Palette)
4. Инспектор Объектов (Object Inspector)
5. Справочник (On-line help)

Дизайнер Форм предназначен для создания интерфейса программы. На форму можно добавлять компоненты (кнопки, окошки ввода текста, переключатели, меню), выбирая их из Палитры компонентов, задавать размер и положение. После расположения компонентов программа еще не готова – кнопки не реагируют на нажатие, введенный текст никак не обрабатывается, переключатели ничего не способны переключить. Чтобы программа заработала, необходимо создать обработчики событий.

Окно редактора кода содержит создаваемые пользователем обработчики событий – процедуры, определяющие, как компонент должен реагировать на определенное событие (наведение курсора мыши на объект, щелчок мышью, двойной щелчок и т.д.). Переключение между Дизайнером форм и Редактором кода – F12.

Палитра компонентов содержит несколько вкладок (Standard, Additional, System и т. д.). На каждой вкладке расположены ряд компонент, обозначенных пиктограммами и именами, появляющимися в виде подсказки. Эти имена являются официальными названиями компонентов. В действительности это названия классов, описывающих компоненты без первой буквы T (например, если класс называется Tbutton, имя будет Button). **Standard.** Большинство компонент на этой вкладке являются аналогами экранных элементов Windows (меню, кнопки, полосы прокрутки), но компоненты обладают также некоторыми удобными дополнительными встроенными возможностями.

Каждый компонент имеет **свойства**, которые устанавливаются в окне *Инспектора Объектов* (*Object Inspector*) на вкладке *Properties*, и связанные с ним обработчики **событий**, которые устанавливаются в окне *Object Inspector* на вкладке *Events*.

Программирование событий

Для *создания* нового проекта необходимо:

- 1) выполнить команду File -> New -> Application for Win32.

Для *сохранения* созданного проекта необходимо:

- 1) создать папку для файлов программы. Т.к. файлов несколько, НИКОГДА не сохранять проект без папки;

- 2) File -> Save Project as. Сохранять в папку. При этом сначала появляется диалог для сохранения модуля, затем – проекта программы. Название проекта и модуля можно менять только при первом сохранении.
- 3) Для сохранения внесенных изменений: File -> Save all.

После сохранения в папке появляются несколько файлов - *.dpr – файл проекта, *.pas – файл модуля, *.dfm – файл формы.

Для **открытия** проекта необходимо:

- 1) File -> Open (File -> Open Project).
- 2) При открытии выделить файл проекта (*.dpr).

Запустить программу на исполнение можно 3 способами:

- 1) F9;
- 2) Run -> Run;
- 3)  на панели инструментов.



TLabel (расположен на вкладке Standard) служит для отображения текста на экране.

Свойства класса TLabel		
<i>Название</i>	<i>Тип</i>	<i>Описание</i>
Caption	String	текст надписи
Autosize	Boolean	определяет возможность подбора ширины компонента автоматически (при true) или вручную (при false)
Font		группа свойств, определяющих параметры шрифта надписи
Top	Integer	координата сверху
Left	Integer	координата слева
Color	Color	цвет фона

Обработчики событий	
<i>Название</i>	<i>Описание</i>
OnMouseMove	Действие происходит при перемещении курсора мыши над объектом



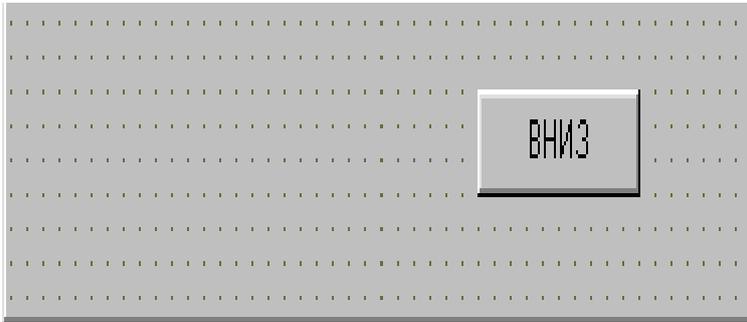
TButton (расположен на вкладке Standard) - кнопка с текстовой надписью.

Свойства класса TButton		
<i>Название</i>	<i>Тип</i>	<i>Описание</i>
Caption	String	текст на поверхности кнопки
Height	Integer	высота компонента
Width	Integer	ширина компонента
Top	Integer	координата сверху
Left	Integer	координата слева

Обработчики событий	
Название	Описание
OnClick	Действие щелчке по объекту

ЗАДАНИЕ.

Создать проект следующего вида:



При щелчке мышью по кнопке «ВВЕРХ» (метод *OnClick*) надпись « - это сила!» должна перемещаться вверх на 10 пикселей (свойство *Top*), а при щелчке по кнопке «ВНИЗ» - на 10 пикселей вниз. При наведении курсора мышки на надпись (метод *OnMouseMove*) она должна стать невидимой (свойство *Visible*).

ВЫПОЛНЕНИЕ ЗАДАНИЯ

- 1) Создать новый проект (команда File -> New -> Application). Сохранить его (File -> Save Project as) в папке Lab1.
- 2) Разместить на форме компоненты: Label и два Button.
- 3) Изменить свойства *Caption* у обоих компонентов и *Font* у компонента Label.
- 4) Создать обработчики событий для кнопок (действие должно происходить при щелчке по кнопке – событие *OnClick*, для создания обработчика события нужно дважды щелкнуть по кнопке на форме).

Для кнопки ВВЕРХ:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
Label1.Top:=Label1.Top-10;
end;
```

Для кнопки ВНИЗ:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
Label1.Top:=Label1.Top+10;
end;
```

- 5) Сохраните проект и проверьте его работу.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ.

Создать проект следующего вида:



При щелчке мышью по кнопке «ВЛЕВО» (метод `OnClick`) надпись «Я люблю !» должна перемещаться влево на 10 пикселей (свойство `Left`), а при щелчке по кнопке «ВПРАВО» - на 10 пикселей вправо. При наведении курсора мышки на надпись (метод `OnMouseMove`) она должна поменять цвет на красный (свойство `Font.color`).

КОНТРОЛЬНЫЕ ВОПРОСЫ:

- 1) Перечислите основные составные части среды программирования и их назначение.
- 2) Как создать новый проект?
- 3) Как сохранить созданный проект?
- 4) Почему сохранять проект можно только в отдельную папку?
- 5) Какие файлы создаются при сохранении проекта?
- 6) Как открыть созданный ранее проект?
- 7) Как запустить проект на исполнение?
- 8) Назовите назначение, основные свойства и методы компонента `Label`.
- 9) Назовите назначение, основные свойства и методы компонента `Button`.

Лабораторная работа № 2

Компонент Edit

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ



TEdit (расположен на вкладке Standard)

Компонент предназначен для ввода и редактирования одной строки текста.

Свойства класса TEdit		
<i>Название</i>	<i>Тип</i>	<i>Описание</i>
MaxLength	Integer	Ограничивает максимально допустимое количество символов в вводимой строке.
ReadOnly	Boolean	Возможность редактирования.
Text	String	Строка текста.
SelText	String	Выделенный текст.
SelStart	Integer	Номер первого символа в выделенном тексте.
SelLength	Integer	Длина выделенного текста.

Методы класса TEdit		
<i>Название</i>	<i>Тип</i>	<i>Описание</i>
Clear	procedure	Удаляет строку из поля ввода.
CopyToClipboard	procedure	Копирует выделенный текст в буфер.
PasteFromClipboard	procedure	Вставляет текст из буфера в позицию курсора или вместо выделенного текста.
CutToClipboard	procedure	Копирует выделенный текст в буфер, а затем удаляет его из поля ввода.

StrToInt

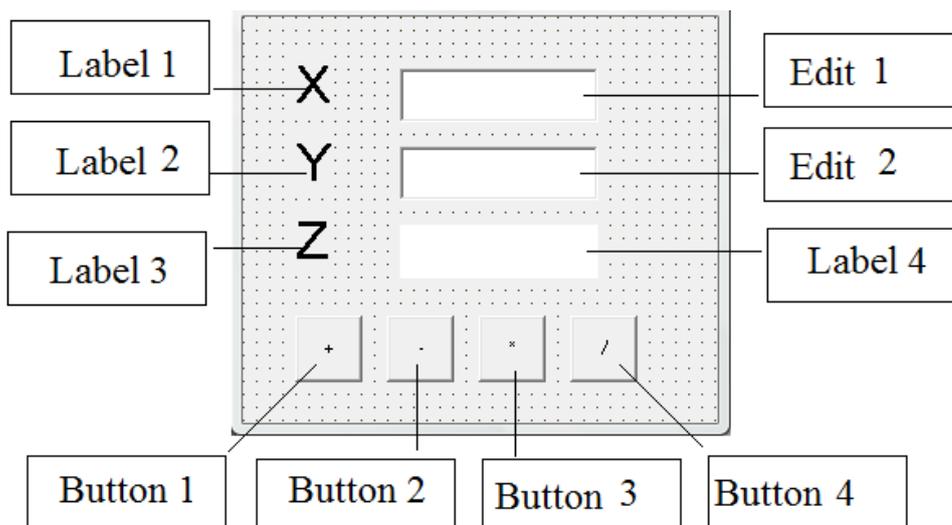
IntToStr

StrToFloat

FloatToStr

ЗАДАНИЕ.

Создать проект калькулятора. Запрограммировать обработчики событий нажатий на кнопки. Предусмотреть деление на ноль.



ВЫПОЛНЕНИЕ ЗАДАНИЯ:

- 1) Создать новый проект (команда File -> New -> Application). Сохранить его (File -> Save Project as) в папке Lab2.
- 2) Расположить на форме указанные компоненты.
- 3) Изменить свойства компонентов:

Название компонента	Свойство	Значение
Label 1	Caption	X
	Font.Size	24
Label 2	Caption	Y
	Font.Size	24
Label 3	Caption	Z
	Font.Size	24
Label 4	Caption	
	AutoSize	False
	Color	clWhite
Edit 1	Text	
Edit 2	Text	
Button 1	Caption	+
Button 2	Caption	-
Button 3	Caption	*
Button 4	Caption	/

- 4) В событиях OnClick каждой кнопки опишите соответствующие действия, напри-

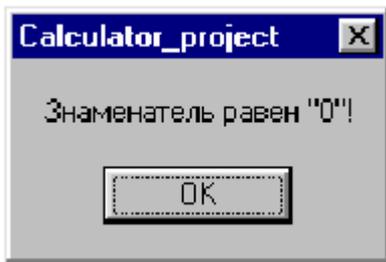
мер, для кнопки  код события должен выглядеть следующим образом:

```

procedure TForm1.Button4Click(Sender: TObject);
begin
  If Edit2.Text='0' Then ShowMessage('Знаменатель равен "0"!')
  Else Label4.Caption:=FloatToStr (StrToFloat (Edit1.Text) / StrToFloat (Edit2.Text) );
end;

```

Обратите внимание, что в данном действии осуществляется проверка деления на ноль. Оператор ShowMessage выдает сообщение, представленное на рисунке в случае, когда Y равен "0".



Для кнопки код события должен выглядеть следующим образом:

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
Label4.Caption:=FloatToStr(StrToFloat(Edit1.Text)+StrToFloat(Edit2.Text));  
end;
```

Для остальных кнопок запишите обработчики событий аналогично примеру.

5) Сохраните проект и проверьте его работу.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ.

- 1) Создайте кнопку для нахождения значений функции $Z=2X+Y^2$.
- 2) Создайте кнопку для нахождения значений функции $Z=X!$ При этом в поле Y копируйте значение X.

Лабораторная работа №3

Компоненты Memo, CheckBox, RadioGroup

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ



TMemo (расположен на вкладке Standard) - служит для ввода многострочного текста до 32 Kb.

Свойства класса TMemo		
Название	Тип	Описание
Lines	TStrings	Текст в поле

Методы класса TMemo	
Название	Описание
Clear	Удаляет текст из поля ввода.



TRadioGroup (расположен на вкладке Standard)

Компонент представляет собой панель с расположенными на ней радиокнопками. Только одна из радиокнопок может быть выделена.

Свойства класса TRadioGroup		
Название	Тип	Описание
Items	TStrings	Список названий радиокнопок.
ItemIndex	Integer	Порядковый номер выделенной радиокнопки.



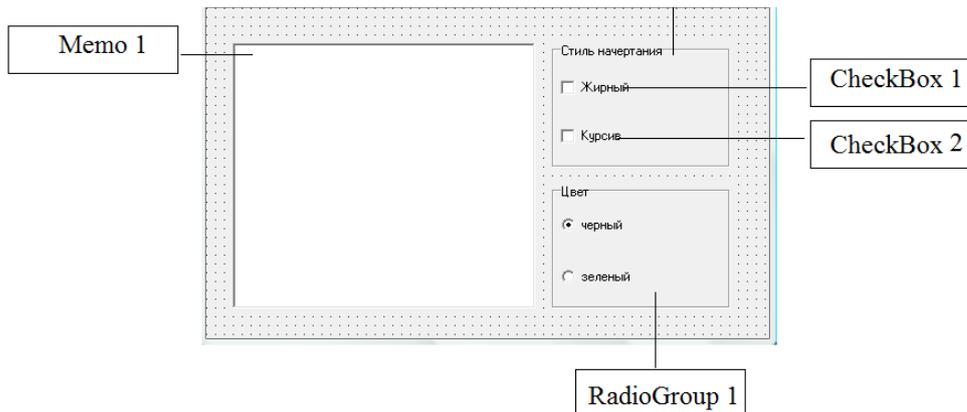
TCheckBox (расположен на вкладке Standard)

Компонент реализует элемент управления – флажок.

Свойства класса TCheckBox		
Название	Тип	Описание
Checked	Boolean	Определяет, установлен флажок или сброшен.
AllowedGrey	Boolean	Определяет, может ли флажок находиться в третьем состоянии – закрашенным в серый цвет.
State	TCheckBox State	Состояние флажка: cbUnchecked – сброшен, cbChecked – установлен, cbGrayed – закрашен в серый цвет.

ЗАДАНИЕ.

Создать проект, позволяющий форматировать текст, вводимый и отображаемый в компоненте Memo.



ВЫПОЛНЕНИЕ ЗАДАНИЯ

- 1) Создать новый проект (команда File -> New -> Application). Сохранить его (File -> Save Project as) в папке Lab3.
- 2) Расположите на форме указанные компоненты.
- 3) Задайте свойства компонентов:

Компонент	Свойство	Значение
Form1	Caption	Форматирование текста
Memo1	Lines	
GroupBox1	Caption	Стиль начертания
CheckBox1	Caption	жирный
CheckBox2	Caption	курсив
RadioGroup1	Caption	Цвет
	Items	черный зеленый
	ItemIndex	0

- 4) Создать процедуру, позволяющую установить выбранный стиль шрифта при щелчке по CheckBox1:

```

procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  if CheckBox1.Checked then Memo1.Font.Style:=[fsbold];
  if CheckBox2.Checked then Memo1.Font.Style:=[fsItalic];
  if (CheckBox1.Checked) and (CheckBox2.Checked) then
    Memo1.Font.Style:=[fsbold,fsItalic];
  if not(CheckBox1.Checked) and not(CheckBox2.Checked) then Memo1.Font.Style:=[];
end;

```

Обратите внимание, что при щелчке по CheckBox2 должны проверяться те же самые условия, поэтому обработчик события будет аналогичным:

```

procedure TForm1.CheckBox2Click(Sender: TObject);
begin
  if CheckBox1.Checked then Memo1.Font.Style:=[fsbold];
  if CheckBox2.Checked then Memo1.Font.Style:=[fsItalic];
  if (CheckBox1.Checked) and (CheckBox2.Checked) then
    Memo1.Font.Style:=[fsbold,fsItalic];
  if not(CheckBox1.Checked) and not(CheckBox2.Checked) then Memo1.Font.Style:=[];
end;

```

6) Создать процедуру, позволяющую установить выбранный цвет:

```
procedure TForm1.RadioGroup1Click(Sender: TObject);
begin
  if RadioGroup1.ItemIndex=0 then Memo1.Font.Color:=clBlack
  else If RadioGroup1.ItemIndex=1 then Memo1.Font.Color:=clGreen;
end;
```

7) Сохраните проект и проверьте его работу.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

- 1) Добавить ещё один CheckBox который будет отвечать за подчеркнутый стиль шрифта.
- 2) Добавить ещё одну кнопку в RadioGroup для синего цвета.
- 3) Добавить ещё один компонент, чтобы пользователь мог устанавливать размер шрифта (10, 12, 14 или 16). Используйте для этого компонент **RadioGroup**.

Лабораторная работа № 4

Компонент StringGrid

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ



TStringGrid (расположен на вкладке Additional)

Компонент расположен на вкладке Additional. Предназначен для представления данных в табличном виде. В каждую ячейку таблицы можно записать строку текста. В верхней и в левой части таблицы имеется фиксированная область, которая не участвует в прокрутке.

Свойства класса TStringGrid		
Название	Тип	Описание
Col	LongInt	Индекс текущего столбца.
Row	LongInt	Индекс текущей строки.
ColCount	LongInt	Количество столбцов в таблице.
ColWidths[Index]	Integer	Ширина столбца с индексом Index.
EditorMode	Boolean	Возможность редактирования ячеек.
FixedColor	TColor	Цвет фиксированной области.
FixedCols	Integer	Количество зафиксированных столбцов.
FixedRows	Integer	Количество зафиксированных строк.
RowCount	LongInt	Количество строк в таблице.
RowHeight[Index]	LongInt	Высота строки с индексом Index.
Cells[ACol, ARow]	String	Содержимое ячейки с координатами (ACol, ARow).
Cols[Index]	TStrings	Столбец с индексом Index.
Rows[Index]	TStrings	Строка с индексом Index.
Objects[ACol, ARow]	TObject	Объект, связанный с ячейкой (ACol, ARow).
DefaultColWidth	Integer	Значение ширины столбца по умолчанию.
DefaultRowHeight	Integer	Значение высоты строки по умолчанию.
ScrollBars	TScrollStyle	Полосы прокрутки: ssNone – отсутствуют, ssHorizontal – горизонтальная, ssVertical – вертикальная, ssBoth – обе.

ЗАДАНИЕ. Сортировка линейного массива методом простого выбора

Создать в среде проект, позволяющий сформировать массив заданной размерности (от 2 до 20 элементов) случайным образом и отсортировать его по возрастанию.



ВЫПОЛНЕНИЕ ЗАДАНИЯ

- 1) Создайте новый проект (команда File -> New -> Application). Сохраните его (File -> Save Project as) в папке Lab4.
- 2) Разместите на форме компоненты StringGrid1, StringGrid2, SpinEdit1 (вкладка Samples), Label1, Label2, Label3, Button1, Button2.
- 3) Задайте свойства компонентов:

Компонент	Свойство	Значение
Form1	Caption	Сортировка массива
Label1	Caption	Введите размерность массива
Label2	Caption	Исходный массив
Label3	Caption	Отсортированный массив
Button1	Caption	Формирование массива
Button2	Caption	Сортировка
StringGrid1, StringGrid2	ColCount	10
	RowCount	1
	FixedCols	0
	FixedRows	0
	DefaultColWidth	35
SpinEdit1	MinValue	2
	MaxValue	20

- 4) В раздел Var добавьте описание глобальных переменных a, i, j, t, n:

```

public
  { Public declarations }
end;

var
  Form1: TForm1;
  a : array[0..19] of integer;
  i, j, t, n : integer;

implementation

```

- 5) Создайте процедуру для формирования массива (обработчик события OnClick для Button1):

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Randomize;
  n:= SpinEdit1.Value;           {количество элементов массива}
  StringGrid1.ColCount:=n;      {определяем нужное число ячеек в таблице}
  StringGrid2.ColCount:=n;
  For i:= 0 to n-1 do {формируем массив и размещаем его в таблице StringGrid1}
  begin
    a[i]:=Random(100);
    StringGrid1.Cells[i,0]:=IntToStr(a[i]);
  end;
end;
```

- 6) Создайте процедуру для сортировки массива (обработчик события OnClick для Button2):

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  for i:=0 to n-2 do
  for j:=i+1 to n-1 do
  if a[i]>a[j] then
  begin
    t:=a[i];
    a[i]:=a[j];
    a[j]:=t;
  end;
  {вывод отсортированного массива в таблице StringGrid2}
  for i:= 0 to n-1 do
  StringGrid2.Cells[i,0]:=IntToStr(a[i]);
end;
```

- 7) Запустите проект на исполнение и проверьте его работу, задавая различную размерность массива.
8) Сохраните проект и проверьте его работу.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ.

Доработайте данный проект следующим образом:

- 1) разместите на форме компонент RadioGroup, позволяющий выбрать одно из направлений сортировки: по возрастанию или по убыванию.
- 2) разместите на форме компоненты CheckBox1 и CheckBox2. При установке флажка в CheckBox1 должны сортироваться элементы массива с четными индексами. При установке флажка в CheckBox2 должны сортироваться элементы массива с нечетными индексами. При установке обоих флажков должны быть отсортированы все элементы массива.

Лабораторная работа №5

Компоненты MainMenu и компоненты вкладки Dialogs

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ



TMainMenu (расположен на вкладке Standard)

Компонент представляет собой меню, расположенное в верхней строке окна.

Свойства класса TMainMenu		
Название	Тип	Описание
Items	TMenuItem	Элемент меню.
Caption	String	Название пункта меню. Символ "&" перед каким-либо символом названия пункта меню выделяет этот символ подчеркиванием. При нажатии Alt-подчеркнутый символ вызывает данный пункт меню. Для создания разделительной полосы между пунктами меню введите символ "-".
Checked	Boolean	Возможность появления галочки или точки (если входит в состав радиогруппы) слева от названия пункта меню.
Enabled	Boolean	Доступность пункта меню.
Items[Index]	TMenuItem	Подменю.
GroupIndex	Byte	Номер радиогруппы, к которой принадлежит пункт меню.
RadioItem	Boolean	Принадлежность радиогруппе.
Visible	Boolean	Видимость пункта в меню.
ShortCut	Word	Соотносит пункту меню комбинацию клавиш.



TOpenDialog,  **TSaveDialog** (расположены на вкладке Dialogs)

Диалоги открытия и сохранения файлов.

Свойства классов TOpenDialog и TSaveDialog		
Название	Тип	Описание
DefaultExt	String	Расширение по умолчанию, добавляется к выбранному файлу.
FileName	String	Имя последнего файла из списка выбранных в диалоге.
Files	TString	Список файлов, выбранных в диалоге.
Filter	String	Маска отбора файлов в диалоге.
FilterIndex	Integer	Фильтр отбора файлов по умолчанию
InitialDir	String	Каталог, к которому диалог обращается при открытии.
Title	String	Заголовок диалога.

Маска фильтра состоит из двух частей, разделенных вертикальной чертой. Первая часть содержит название фильтра, а вторая – маску файла. Например,

'Модули проекта | *.pas'.

Несколько фильтров разделяются вертикальной чертой. В одном фильтре можно отбирать файлы с различными расширениями. Пример более сложного фильтра:

Графические файлы | *.bmp; *.jpg; *.gif; *.psx | РСХ-файлы | *.psx'.



TFontDialog (расположен на вкладке Dialogs)

Диалог установки параметра шрифта.

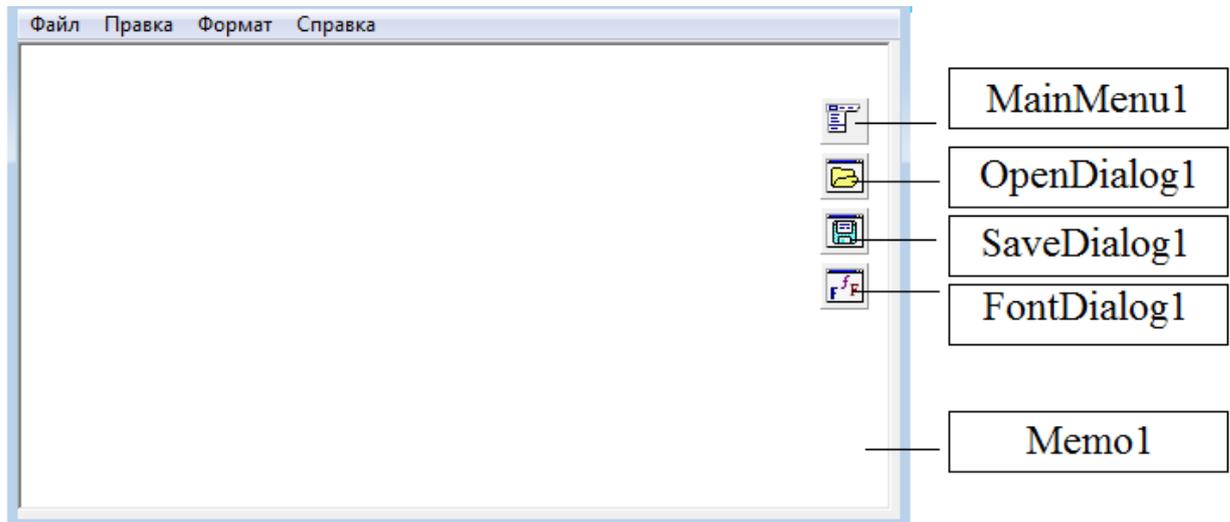
Свойства класса TFontDialog		
Название	Тип	Описание
Font	TFont	Содержит параметры шрифта.

Компоненты диалогов не являются визуальными. Для того чтобы вызвать диалоговое окно, необходимо вызвать метод Execute, после вызова которого появляется диалоговая форма. Если пользователь выбрал один или несколько файлов, или выбрал цвет, или выбрал параметры шрифтов и затем нажал кнопку "ОК", то функция возвращает значение true. Если пользователь нажал кнопку "Cancel", то функция возвращает значение false.

Методы классов диалогов		
Название	Тип	Описание
Execute: Boolean	function	Вызывает окно диалога.

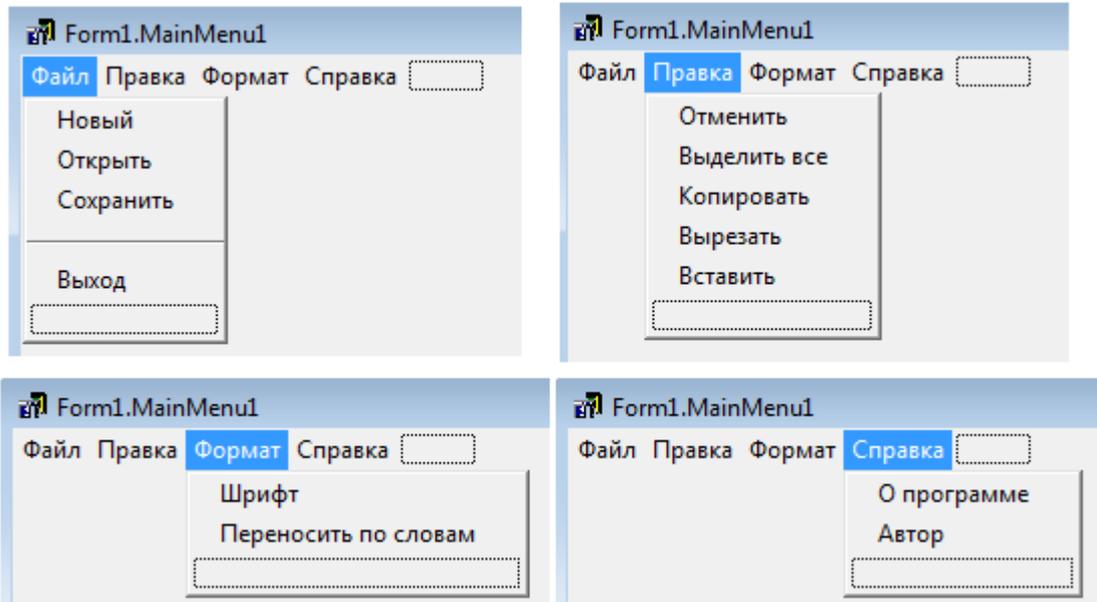
ЗАДАНИЕ.

Создать проект, позволяющий форматировать и сохранять текст, вводимый с клавиатуры, а также открывать ранее сохраненный текст (простейший текстовый редактор).



ВЫПОЛНЕНИЕ ЗАДАНИЯ

- 1) Создайте новый проект (команда File -> New -> Application). Сохраните его (File -> Save Project as) в папке Lab5.
- 2) Расположите на форме указанные компоненты.
- 3) Создайте главное меню следующего вида:



Для создания меню необходимо дважды щелкнуть по значку MainMenu, в появившемся окне ввести название первого пункта меню и нажать Enter. После этого можно вводить пункты меню ниже или правее. Чтобы создать горизонтальную черту, вместо названия нужно ввести тире (-).

У компонента Memo1 свойство Align (расположение) установите в alClient.

- 4) Запрограммируйте кнопку **Новый** (чтобы создать обработчик события для пункта меню, дважды щелкните по нему).

```
procedure TForm1.N5Click(Sender: TObject);
begin
  Memo1.Clear
end;
```

Зaprogramмируйте кнопку **Открыть**.

```
procedure TForm1.N6Click(Sender: TObject);
begin
  if OpenFileDialog1.Execute then
    Memo1.Lines.LoadFromFile(OpenDialog1.FileName)
end;
```

Зaprogramмируйте кнопку **Сохранить**.

```
procedure TForm1.N7Click(Sender: TObject);
begin
  if SaveDialog1.Execute then
    Memo1.Lines.SaveToFile(SaveDialog1.FileName)
end;
```

Зaprogramмируйте кнопку **Выход**.

```
procedure TForm1.N9Click(Sender: TObject);
begin
  close
end;
```

Зaprogramмируйте кнопку **Отменить**.

```
procedure TForm1.N10Click(Sender: TObject);
begin
  Memo1.Undo
end;
```

Запрограммируйте кнопку **Выделить все**.

```
procedure TForm1.N11Click(Sender: TObject);
begin
  Memo1.SelectAll
end;
```

Запрограммируйте кнопку **Копировать**.

```
procedure TForm1.N12Click(Sender: TObject);
begin
  Memo1.CopyToClipboard
end;
```

Запрограммируйте кнопку **Вырезать**.

```
procedure TForm1.N13Click(Sender: TObject);
begin
  Memo1.CutToClipboard
end;
```

Запрограммируйте кнопку **Вставить**.

```
procedure TForm1.N14Click(Sender: TObject);
begin
  Memo1.PasteFromClipboard
end;
```

Запрограммируйте кнопку **Шрифт**.

```
procedure TForm1.N15Click(Sender: TObject);
begin
  if FontDialog1.Execute then
    Memo1.Font:=FontDialog1.Font
end;
```

Запрограммируйте кнопку **Переносить по словам**.

```
procedure TForm1.N16Click(Sender: TObject);
begin
  Memo1.ScrollBars:=ssVertical
end;
```

5) Сохраните проект и проверьте его работу.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

- 1) Запрограммируйте кнопку **О программе** так, чтобы выводилось сообщение «Простейший текстовый редактор».
- 2) Запрограммируйте кнопку **Автор** так, чтобы выводилось сообщение с информацией о вас.
- 3) Добавьте в меню Правка ещё один пункт - **Очистить** и запрограммируйте его так, чтобы текстовое поле очищалось.

Лабораторная работа № 6

Компонент Timer

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ



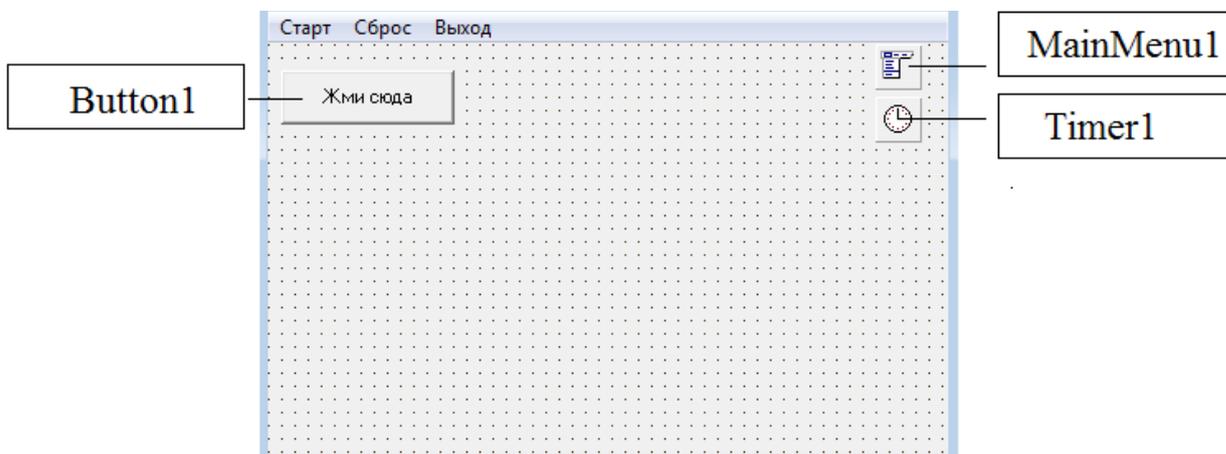
TTimer (расположен на вкладке System)

Компонент реализует функции системного таймера. Это невизуальный компонент. После того как в свойстве Enabled будет установлено значение true, компонент переходит в режим ожидания сообщений от системного таймера. Сообщения от системного таймера приходят через промежутки времени, равные значению, установленному в свойстве Interval. После получения сообщения вызывается событие OnTimer, после обработки которого компонент опять переходит в режим ожидания следующего сообщения от системы. Этот процесс продолжается, пока в свойстве Enabled не будет установлено значение false.

Свойства класса TTimer		
Название	Тип	Описание
Enabled	Boolean	Позволяет получать сообщения от системного таймера.
Interval	Integer	Интервал времени, через который вызывается событие OnTimer. Значение 1000 равняется одной секунде.

ЗАДАНИЕ.

Создать проект «Поймай кнопку». После запуска приложения по форме перемещается кнопка случайным образом. Необходимо «поймать» ее щелчком мыши. После этого выводится сообщение о победе.



ВЫПОЛНЕНИЕ ЗАДАНИЯ

- 1) Создайте новый проект (команда File -> New -> Application). Сохраните его (File -> Save Project as) в папке Lab6.
- 2) Расположите на форме указанные компоненты.
- 3) Создайте главное меню по образцу и установите следующие свойства компонентов:

Компонент	Свойство	Значение
Form1	Caption	Поймай кнопку
Timer1	Interval	800

Button1	Caption	жми сюда
---------	---------	----------

4) Создайте обработчик события по созданию формы (OnCreate):

```

procedure TForm1.FormCreate(Sender: TObject);
begin

    Form1.Height:=350;           {высота формы}
    Form1.Width:=550;           {ширина формы}
    timer1.Enabled:=false;      {таймер выключен}
    button1.Visible:=false;     {кнопка невидима}
    randomize;                   {иницилируем датчик случайных чисел}
end;

```

5) Запрограммируйте пункт меню **Старт**:

```

procedure TForm1.N1Click(Sender: TObject);
begin
    timer1.Enabled:=true;       {таймер включаем}
    button1.Visible:=true;      {кнопка видима}
end;

```

6) Запрограммируйте пункт меню **Сброс**:

```

procedure TForm1.N2Click(Sender: TObject);
begin
    button1.Visible:=false;     {кнопка невидима}
    timer1.Enabled:=false;     {таймер выключен}
end;

```

7) Запрограммируйте выбор пункта меню **Выход**:

```

procedure TForm1.N3Click(Sender: TObject);
begin
    close;
end;

```

8) Запрограммируйте изменение положения кнопки через заданный интервал времени:

```

procedure TForm1.Timer1Timer(Sender: TObject);
begin
    button1.Top:=random(250);
    button1.Left:=random(450);
end;

```

9) Запрограммируйте щелчок мышью по кнопке Button1:

```

procedure TForm1.Button1Click(Sender: TObject);
begin
    timer1.Enabled:=false;      {таймер выключаем}
    showmessage('ПОВЕДА!!!');   {выводим сообщение о победе}
end;

```

10) Сохраните проект и проверьте его работу.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ.

Доработайте проект следующим образом:

поместите на форму компонент ScrollBar1 (вкладка Standart) и предусмотрите возможность с его помощью изменять скорость перемещения кнопки.

Справка: ScrollBar

Лабораторная работа № 7

Графические возможности .
Компоненты Image, Shape, OpenPicrureDialog,
SavePictureDialog, ColorBox, Combobox.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ



TImage (расположен на вкладке Additional)

Компонент представляет собой картинку.

Свойства класса TImage		
Название	Тип	Описание
Autosize	Boolean	Определяет, будут ли размеры компонента изменяться в соответствии с размерами рисунка.
Picture	TPicture	Рисунок.
Stretch	Boolean	Определяет, будут ли размеры рисунка изменяться в соответствии с размерами компонента.
Transparent	Boolean	Определяет, будет ли фон рисунка прозрачным.



TShape (расположен на вкладке Additional) - служит для отображения простейших графических объектов на форме: окружность, квадрат и т.п. (Shape).



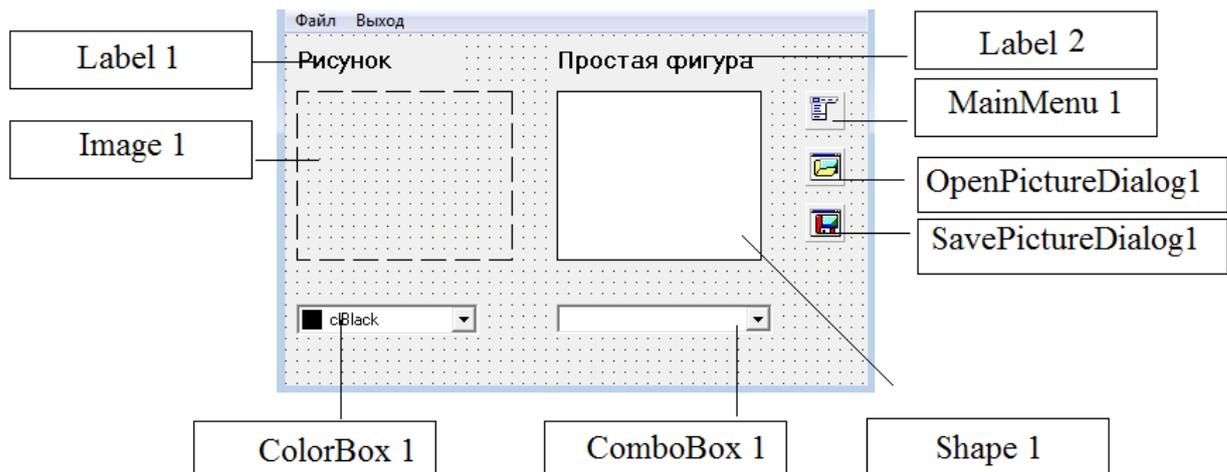
TComboBox (расположен на вкладке Standard)

Компонент представляет собой раскрывающийся список.

Свойства класса TComboBox		
Название	Тип	Описание
Items	TStrings	Строки списка.
ItemIndex	Integer	Номер выбранной строки.
Style	TComboBoxStyle	Стиль: csDropDown – раскрывающийся список и поле ввода, csSimple – поле ввода с обычным списком, csDropDownList, – раскрывающийся список без поля ввода.

ЗАДАНИЕ.

Создать проект, позволяющий просматривать и редактировать рисунки, а также просматривать простейшие геометрические фигуры

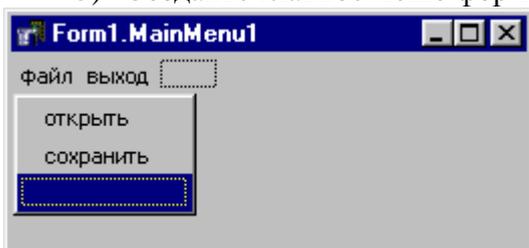


ВЫПОЛНЕНИЕ ЗАДАНИЯ

- 1) Создайте новый проект (команда File -> New -> Application). Сохраните его (File -> Save Project as) в папке Lab7.
- 2) Разместите на форме указанные компоненты. Установите следующие свойства компонентов

Компонент	Свойство	Значение
Label1	Caption	Рисунок
Label2	Caption	Простая фигура
ComboBox1	Items	квадрат круг
Image1	AutoSuze	true
	Stretch	false

- 3) Создайте главное меню формы



- 4) Запрограммируйте выбор пункта меню **Открыть**:

```
procedure TForm1.N2Click(Sender: TObject);
begin
  if OpenPictureDialog1.Execute then Image1.Picture.LoadFromFile(OpenPictureDialog1.FileName);
end;
```

- 5) Запрограммируйте выбор пункта меню **Сохранить**:

```
procedure TForm1.N3Click(Sender: TObject);
begin
  if SavePictureDialog1.Execute then Image1.Picture.SaveToFile(SavePictureDialog1.FileName);
end;
```

- 6) Запрограммируйте выбор пункта меню **Выход**:

```
procedure TForm1.N4Click(Sender: TObject);
begin
  Close;
end;
```

- 7) Запрограммируйте выбор цвета для рисования с помощью компонента ColorBox1 (событие onChange):

```
procedure TForm1.ColorBox1Change(Sender: TObject);
begin
  Image1.Canvas.Pen.Color:=ColorBox1.Selected;
end;
```

- 8) Запрограммируйте установку графического указателя при нажатии клавиши мыши в пределах компонента Image1 (событие onMouseDown):

```
procedure TForm1.Image1MouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
  Image1.Canvas.MoveTo(X, Y);
end;
```

- 9) Запрограммируйте рисование линии при нажатии левой клавиши мыши и ее перемещении по компоненту Image1 (событие onMouseMove):

```
procedure TForm1.Image1MouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
begin
  if ssLeft in Shift then Image1.Canvas.LineTo(X, Y);
end;
```

- 10) Запрограммируйте выбор простой геометрической фигуры с помощью компонента ComboBox1 (событие onChange):

```
procedure TForm1.ComboBox1Change(Sender: TObject);
begin
  case ComboBox1.ItemIndex of
    0: Shape1.Shape:=stSquare;
    1: Shape1.Shape:=stCircle;
  end;
end;
```

- 11) Сохраните проект и проверьте его работу.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Доработайте проект следующим образом:

- 1) добавьте в список ComboBox1 новые пункты: **прямоугольник** и **эллипс** и предусмотрите возможность изображения этих фигур;
- 2) запрограммируйте закраску фигуры Shape1 цветом, выбранном в компоненте ColorBox1;
- 3) добавьте на форму две кнопки Button1 и Button2. При щелчке мышью по Button1 размеры компонента Shape1 увеличиваются вдвое, а при щелчке по Button2 – уменьшаются вдвое.

Лабораторная работа №8

Классы и модули

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Модули и программы

Приложение создается из файлов двух разных видов. Это один файл программы (*.dpr) и один или несколько модулей (*.pas).

Файл программы является основным – он связывает воедино все модули. Просмотреть содержимое файла программы можно, выполнив команду Project -> View Source. Как правило, код программы генерируется автоматически, поэтому многие начинающие программисты его не замечают.

Модули можно считать вторичными файлами, к которым обращается основная часть приложения – программа.

Модули

Приложения интенсивно используют модули. За каждой формой скрывается соответствующий ей модуль. Однако модули не обязаны иметь соответствующие формы.

Модуль состоит из двух основных разделов - interface, где объявлено все, что доступно для других модулей (процедуры, функции, переменные) и раздела implementation с реальным кодом процедур и функций.

Кроме того, модуль может иметь два необязательных раздела: initialization с некоторым кодом запуска, который выполняется при загрузке в память программы, использующей данный модуль, и finalization, который выполняется при завершении программы.

В начале раздела interface может находиться предложение uses. Оно указывает к каким другим модулям мы должны получить доступ из раздела interface текущего модуля. Если же на другие модули необходимо сослаться из кода подпрограмм и методов, вы должны добавить новое предложение uses в начале раздела implementation.

В интерфейсе модуля можно объявить несколько различных элементов, в том числе процедуры, функции, глобальные переменные и типы данных. Также можно поместить в модуль класс.

Чтобы создать новый не относящийся к форме модуль, выберите команду File/New и отметьте на странице New появившегося окна Object Repository элемент Unit.

Классы и сокрытие информации

Что такое класс

Класс может содержать сколько угодно данных и любое количество методов. Однако для соблюдения всех правил объектно-ориентированного подхода данные должны быть скрыты, или инкапсулированы, внутри использующего их класса. Использование метода для получения доступа к внутреннему представлению объекта уменьшает риск возникновения ошибочных ситуаций и позволяет автору класса модифицировать внутреннее представление в будущих версиях. В Object Pascal имеются две различные конструкции, подразумевающие инкапсуляцию, защиту и доступ к переменным: классы и модули. С классами связаны некоторые специальные ключевые слова – спецификаторы доступа:

◆ private – элементы интерфейса класса видны только в пределах модуля, в котором определен класс. Вне этого модуля private-элементы не видны и недоступны. Если в одном модуле создано несколько классов, то все они видят private-разделы друг друга.

◆ Раздел `public` не накладывает ограничений на область видимости перечисленных в нем полей, методов и свойств. Их можно вызывать в любом другом модуле программы.

◆ Раздел `published` не ограничивает область видимости, однако в нем перечислены свойства, которые должны быть доступны не только на этапе исполнения, но и на этапе конструирования программы. Раздел `published` используется при разработке компонентов. Без объявления раздел считается объявленным как `published`. Такой умалчиваемый раздел располагается в самом начале объявления класса любой формы и продолжается до первого объявленного раздела. В раздел `published` среда помещает описание вставляемых в форму компонентов. Сюда не нужно помещать собственные элементы или удалять элементы, вставленные средой.

◆ Раздел `protected` доступен только методам самого класса, а также любым потомкам, независимо от того, находятся ли они в том же модуле или нет.

Порядок следования разделов может быть любой, любой раздел может быть пустым.

Информация о типе на этапе выполнения

Правило языка `Object Pascal` о совместимости типов для классов-потомков позволяет вам использовать класс-потомок там, где ожидается класс - предок, обратное невозможно. Теперь предположим, что класс `Dog` содержит функцию `Eat`, которая отсутствует в классе `Animal`.

Если переменная `MyAnimal` ссылается на объект типа `Dog`, вызов этой функции должен быть разрешен. Но если вы попытаетесь вызвать эту функцию, а переменная ссылается на объект другого класса, возникнет ошибка. Поскольку компилятор не способен определить, будет ли значение правильным на этапе выполнения, то, делая явное приведение типов, мы рискуем вызвать опасную ошибку этапа выполнения программы. Для решения данной проблемы можно воспользоваться некоторыми подходами, основанными на системе RTTI. Каждый объект знает свой тип и своего предка и может получить эту информацию с помощью операции `is`. Параметрами операции `is` являются объект и тип:

```
if MyAnimal is Dog then ...
```

Выражение `is` становится истинным, только если в настоящее время объект `MyAnimal` имеет тип `Dog` или тип потомка от `Dog`. Другими словами, это выражение приобретает значение `True`, если вы можете без риска присвоить объект (`MyAnimal`) переменной заданного типа данных (`Dog`).

Такое прямое приведение можно выполнить так:

```
if MyAnimal is Dog then  
  MyDog := Dog (MyAnimal) ;
```

То же действие можно выполнить напрямую с помощью другой операции RTTI – `as`. Мы можем написать так:

```
MyDog := MyAnimal as Dog ;  
Text := MyDog. Eat ;
```

Если мы хотим вызвать функцию `Eat`, можно использовать и другую форму записи:

```
(MyAnimal as Dog) . Eat ;
```

Результатом выражения будет объект с типом данных класса `Dog`, поэтому вы можете применить к нему любой метод этого класса.

Приведение с операцией `as` отличается от традиционного приведения тем, что в случае несовместимости типа объекта с типом, к которому вы пытаетесь его привести, порождается исключение `EInvalidCast`. Чтобы избежать этого исключения, используйте операцию `is` и в случае успеха делайте простое приведение:

```
if MyAnimal is Dog then
  (Dog (MyAnimal)) . Eat ;
```

ЗАДАНИЕ.

Создать программу, позволяющую выводить информацию о еде и голосе животных собаки и кошка. Описание животных оформить в виде класса.

ВЫПОЛНЕНИЕ ЗАДАНИЯ

- 1) Создать новый проект (File/New Application).
- 2) Сохранить проект в папке Lab8 (Unit1.pas под новым именем Main.pas, а Project1.dpr под новым именем Lab.dpr).
- 3) Открыть модуль, не связанный с формой (File/New/Unit), и поместить в него три класса:
 - ◆ Класс `Animal`, который содержит в разделе `public` объявление конструктора `Create` и объявление метода-функции: `Voice` – звук, издаваемый животным. Тип результата возвращаемого функцией, – `string`. Метод `Voice` объявить виртуальным и абстрактным. В разделе `private` класса определить переменную `Kind: string`.

```
unit Unit1;

interface

type
  TAnimal=class
  private
    Kind:string;
  public
    constructor Create;
    function Voice: string; virtual; abstract;
  end;
```

- ◆ Класс `Dog` объявить потомком класса `Animal`: `TDog = class (TAnimal)`. В разделе `public` этого класса объявить конструктор и методы `Voice` и `Eat`. Метод `Eat` типа `string` объявить виртуальным (пища животного).

```
TDog=class(TAnimal)
  public
    constructor Create;
    function Voice: string; override;
    function Eat: string; virtual;
  end;
```

- ◆ Класс `Cat` объявить потомком класса `Animal`: `TCat = class (TAnimal)`. Раздел `public` класса содержит те же определения, что и соответствующий раздел класса `Dog`.

```
TCat=class(TAnimal)
public
    constructor Create;
    function Voice: string; override;
    function Eat: string; virtual;
end;
```

В реализациях конструктора каждого класса (раздел Implementation) переменной Kind присваивается имя соответствующего животного, например, для класса Animal:

```
implementation
constructor TAnimal.Create;
begin
    kind:='An Animal';
end;
```

Для конструкторов классов TDog и TCat код будет аналогичен (допишите его сами).

В реализациях методов Voice возвращается звук, издаваемый животным, например, для класса TDog (для класса TCat – аналогично):

```
function TDog.voice;
begin
    Result:='gav';
end;
```

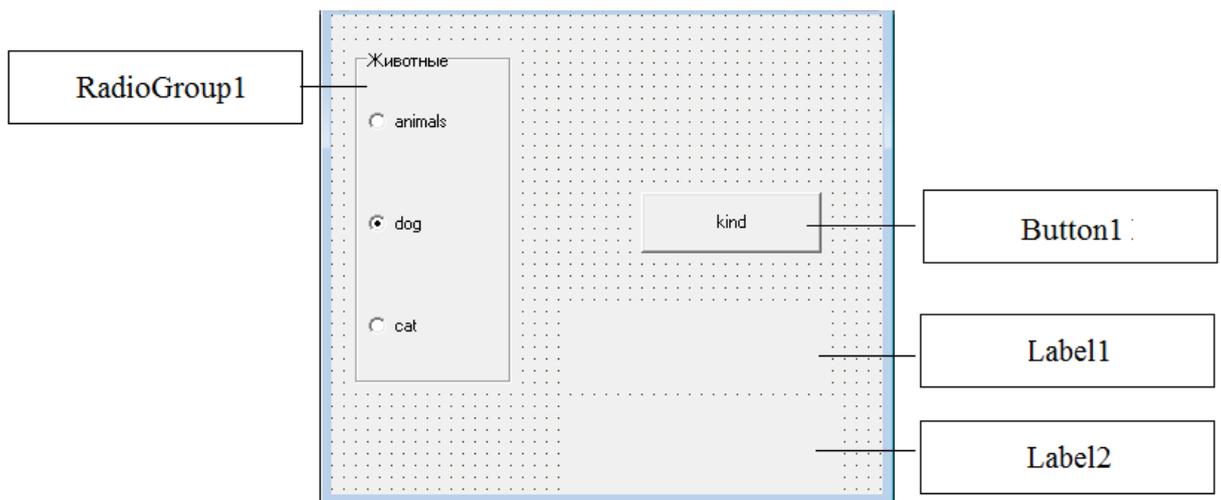
Переменная Result возвращает значение в функцию.

Поскольку в классе TAnimal метод Voice объявлен абстрактным, его реализация не требуется.

В реализациях методов Eat возвращается название пищи, которой питается соответствующее животное:

```
function TDog.eat;
begin
    Result:='кость';
end;
```

- 4) Сохранить созданный модуль.
- 5) На форме расположить компоненты, задать свойства аналогично рисунку (обратите внимание, что в RadioGroup второй пункт выбран по умолчанию – свойство ItemIndex):



Выбору одной из кнопок опций будет соответствовать выбор животного.

При нажатии кнопки команды надписи должны отобразить звук, издаваемый животным, и его пищу.

Определить в классе формы private-переменную MyAnimal:

```

type
  TForm1 = class(TForm)
    RadioGroup1: TRadioGroup;
    Button1: TButton;
    Label1: TLabel;
    Label2: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure RadioGroup1Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    MyAnimal: TAnimal; { Private declarations }
  public
    { Public declarations }
  end;

```

- 6) Записать код для обработчика события OnCreate формы, где создается объект типа Dog, на который ссылается переменная MyAnimal:

```

procedure TForm1.FormCreate(Sender: TObject);
begin
  MyAnimal := TDog.Create;
end;

```

- 7) В обработчике события OnClick компонента RadioGroup записать код, который удаляет текущий объект и создает новый в зависимости от выбранного пункта:

```

procedure TForm1.RadioGroup1Click(Sender: TObject);
begin
  MyAnimal.Free;
  Case RadioGroup1.ItemIndex of
    0: MyAnimal := TAnimal.Create;
    1: MyAnimal:=TDog.Create;
    2: MyAnimal:=TCat.Create;
  end;
end;

```

- 8) В обработчике события OnClick кнопки записать код, который будет помещать в надписи звук, издаваемый животным и его еду:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Label1.Caption := MyAnimal.Voice;
  if MyAnimal is TDog then Label2.Caption:=(TDog (MyAnimal) ) . Eat else
  if MyAnimal is TCat then Label2.Caption:=(TCat (MyAnimal) ) . Eat else
  Label2.Caption:='';
end;
```



Если вы все сделали правильно, при запуске приложения надписи будут отображать пищу и звук для Dog и Cat и приложение завершит работу по ошибке при выборе Animal. Уберите ключевое слово abstract в объявлении метода Voice и реализуйте его, например, так:

```
function TAnimal.voice;
begin
  Result:='ay';
end;
```

Запустите приложение снова. Посмотрите, что изменилось в работе приложения. Объясните различия.

- 9) Попробуйте использовать метод Eat без приведения типов (без is).
10) Сохраните проект.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ.

Доработайте проект следующим образом:

- 1) дополните каждый класс животных новой характеристикой – среда обитания.
- 2) разработайте два класса потомка от TAnimal, которые будут отображать особенности двух новых животных Fish (рыба) и Bird (птица).
- 3) доработайте форму так, чтобы можно было увидеть все характеристики разработанных классов.